NYC Rideshare Analysis

**Task 1 (15 points): Merging Datasets Task 1(1) Loading Data**:

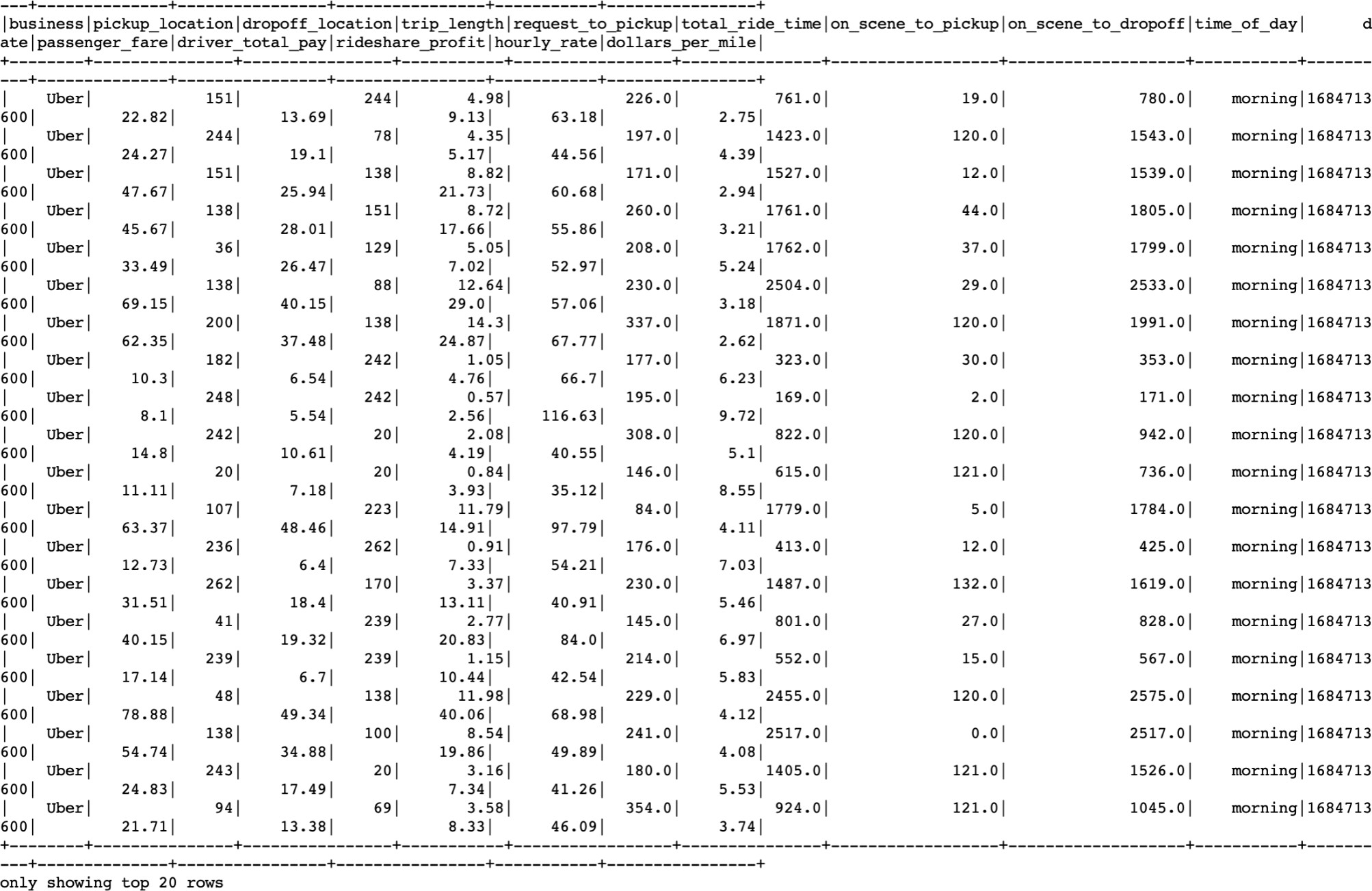
**Explanation:**

I started by loading rideshare\_data.csv and taxi\_zone\_lookup.csv into data frames.

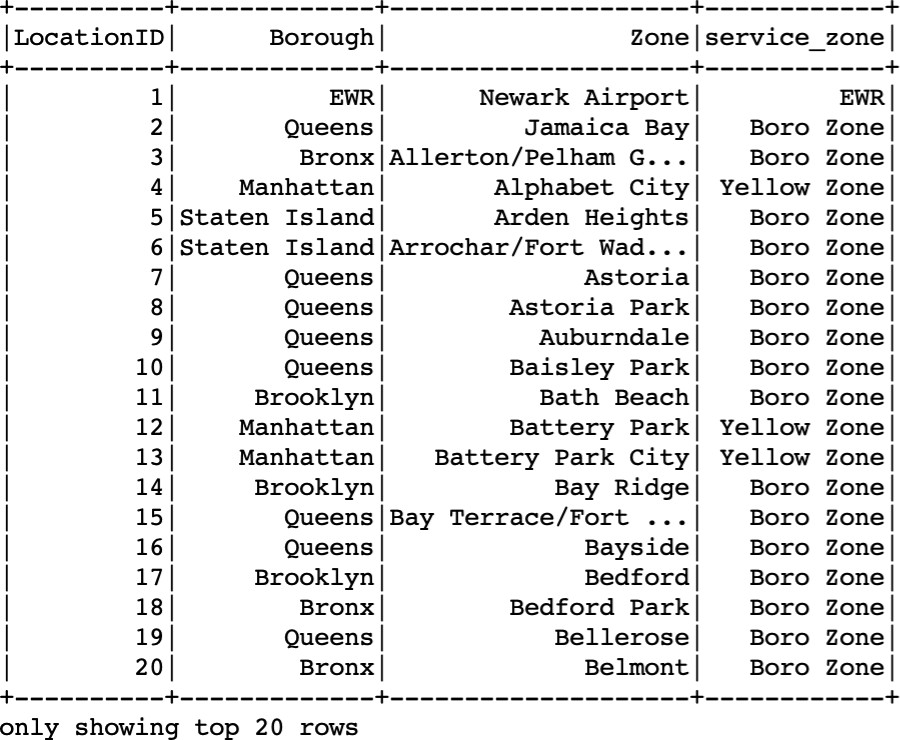
NOTE : (header =True) read the first row as the header for accurately interpreting the dataset structure from the beginning and (inferSchema=True) to infer data types

**Visualization/Screenshots:**

rideshare\_data.csv dataframe:



taxi\_zone\_lookup.csv dataframe:



**Challenges/How I overcome them:**

**Challenges:**

Ensuring correct file paths and understanding the data schema were initial challenges.

**Overcoming Challenges:**

Ensured that the file paths provided were accurate and accessible.I reviewed the data schema to understand column names, data types, and ensure that the file path provided were accurate to ensure spark can read file without any error

**Insight:**

Understanding the structure and content of the datasets is crucial for effective data processing.

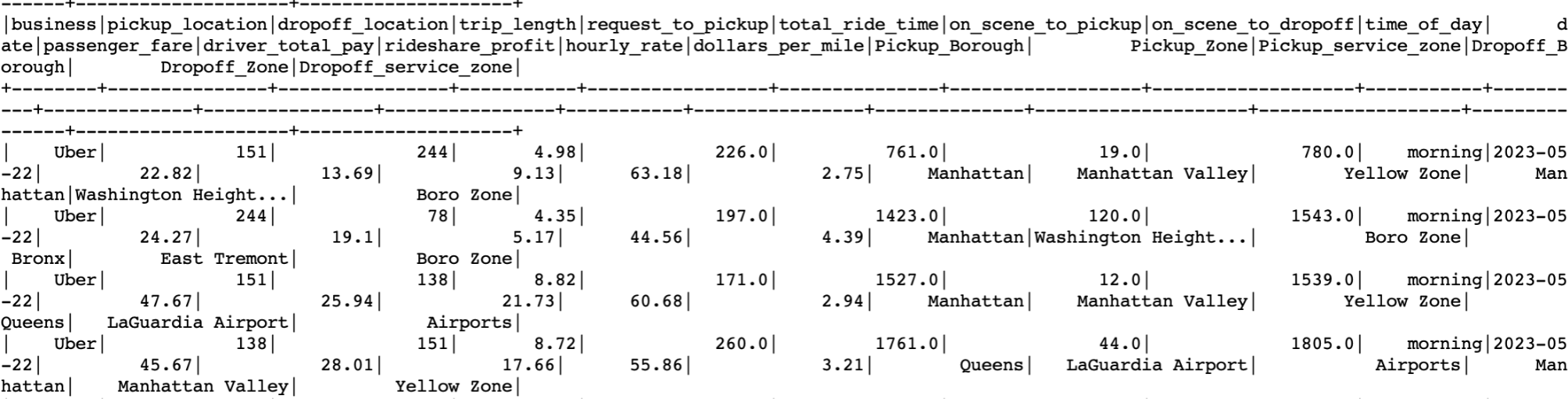
**Task 1(2) Applying Join Operations**:

**Explanation:**

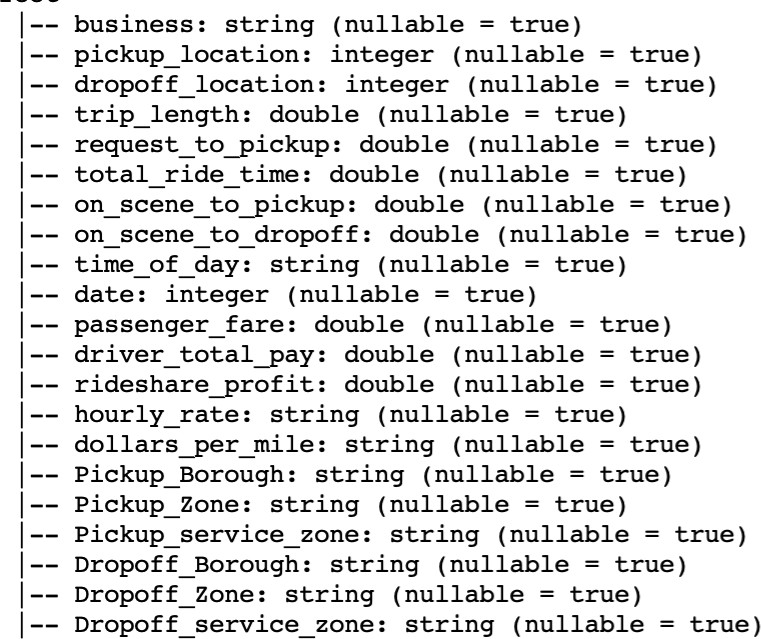
I use the join function(inner joint) and then check the conditions for **pickup\_location** and **dropoff\_location** with **LocationID**. Where I Joined “rideshare\_data” with “ taxi\_zone\_lookup” based on “pickup\_location” and again Joined “rideshare\_data” again with “taxi\_zone\_lookup” based on “dropoff\_location” and renamed columns for clarity and dropped the “LocationID” column from both joins.

**Visualization/Screenshots:**

Dataframe:



Schema :



**Challenges/How I overcome them:**

**Challenges:**

Ensuring correct column names and proper joining conditions.

**Overcoming Challenges:**

Checked the column names and their corresponding values in both datasets to ensure compatibility for joining and applied joining conditions based on common columns.

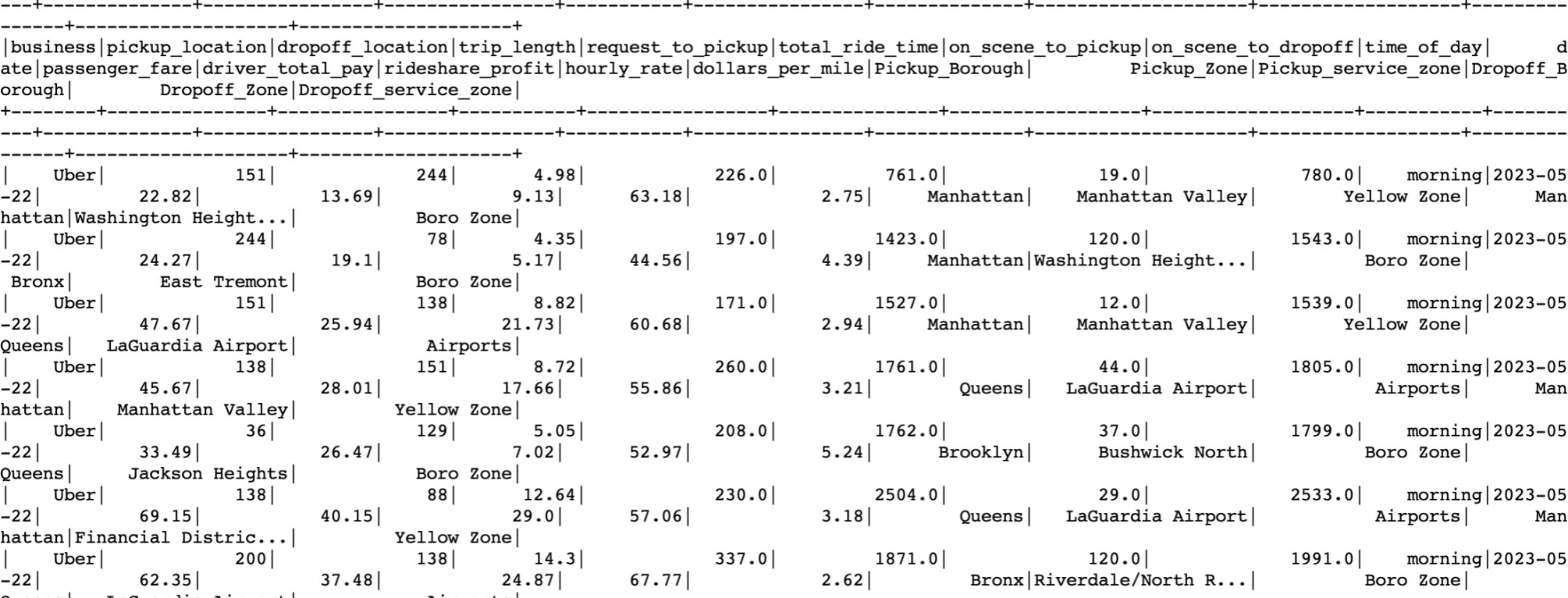
**Insights:**

Joins are essential for combining information from multiple tables and concepts learned while applying joining conditions.

**Task 1(3): Convert the UNIX timestamp to the "yyyy-MM-dd" format Explanation:**

In this step I converted the UNIX timestamp in the date field to a readable "yyyy-MM-dd" date format and used functions like from\_unixtime and date\_format.

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

Understanding the UNIX timestamp format and how to convert it to the desired format with correct usage of Spark DataFrame functions.

**Overcoming Challenges:**

Understanding Timestamp Format and usage of dataframe functions such as from\_unixtime to correctly apply timestamp conversions and using right syntax.

**Insights:**

How a simple timestamp format is essential for more understanding of a given dataset and learning more about different timestamps which are available to work in near future.

**Task 1(4): Count (Give total number of rows):**

**Explanation:**

I printed the updated dataframe's row count and schema using .count() and

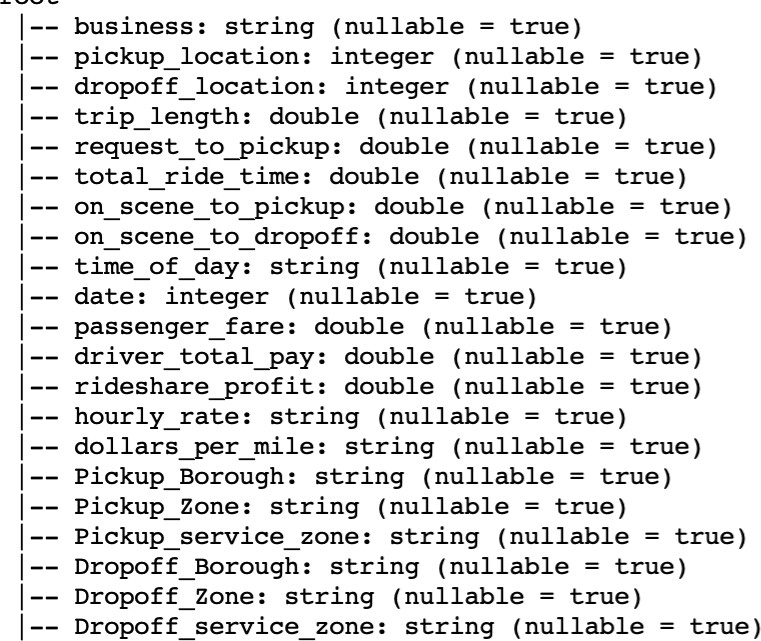
.printSchema() as asked in the question.

**Visualization/Screenshots:**

**Row Count :**



**Schema:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring efficient execution, especially on large datasets like this and Understanding the implications of counting rows in terms of performance and resource usage.

**Overcoming Challenges:**

I overcome my challenge of understand the implications of rows and its impact on performance using right syntax is essential to optimized spark configuration and potential resource consumption to maintain overall performance

**Insights:**

How large a dataset can be and for how many records we are dealing with, which ultimately results in performance and resource usuage.

**—------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

**Task 2 (20 points): Aggregation of Data**

**Task 2(1): Count the number of trips for each business in each month and Draw a Histogram**

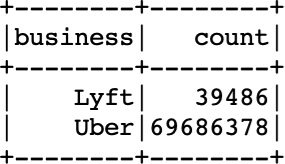
**Explanation:**

The data was grouped by 'business' and 'month' columns using groupBy, and the count of trips was calculated using the .count() function.

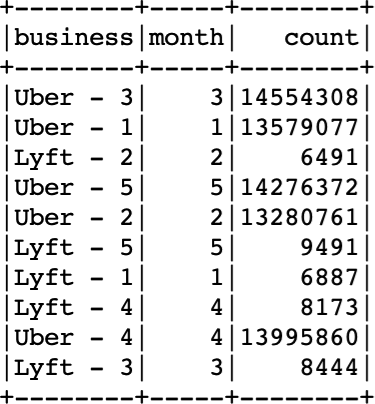
Then, I visualized the results using a histogram where each bar represents the number of trips for each business in each month.

**Visualization/Screenshots:**

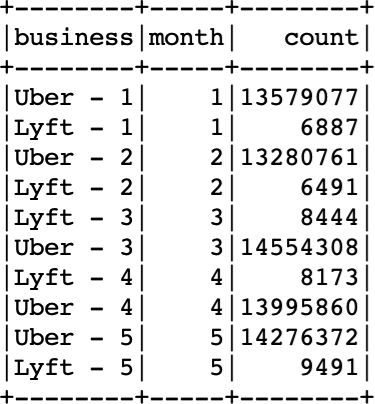
**No of trips( count ) each business:**



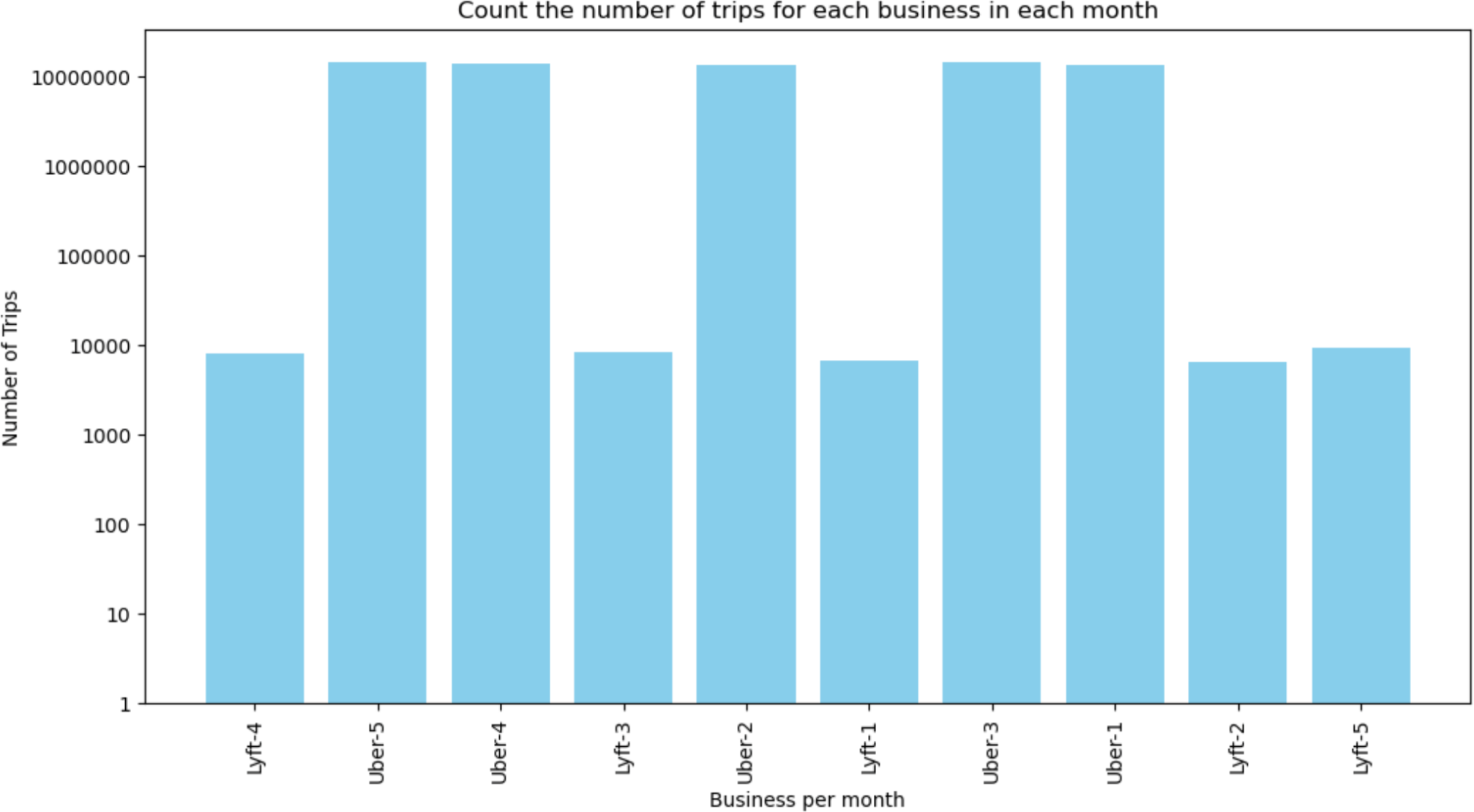
**count(No. Of trips) Business-month:**



**Business-month Sorted by month:**



**Histogram :**



**Challenges/How I overcome them:**

**Challenges:**

Understanding the data structure and grouping requirements and choosing appropriate visualization techniques for presenting the data effectively.

**Overcoming Challenges:**

Thoroughly examine the data schema and sample data to understand column names, data types, and relationships between different datasets. This helps in determining the appropriate columns for grouping and analysis.

**Insights:**

The analysis likely reveals how each business's trip changes over the month providing insights into market dominance

Logarithmic transformation is applied to the 'count' column to normalize the distribution of data, reduce the skewness, and manage large range values which is associated with lyft business.

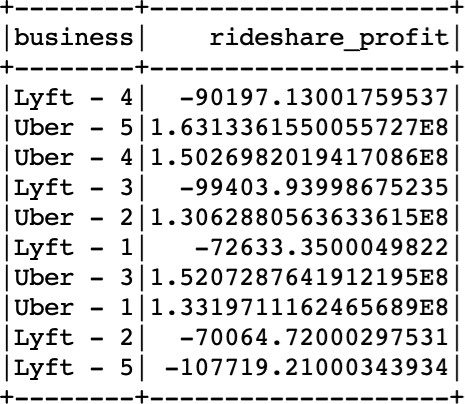
**Task 2(2): Calculate the monthly profit for each business**

**Explanation:**

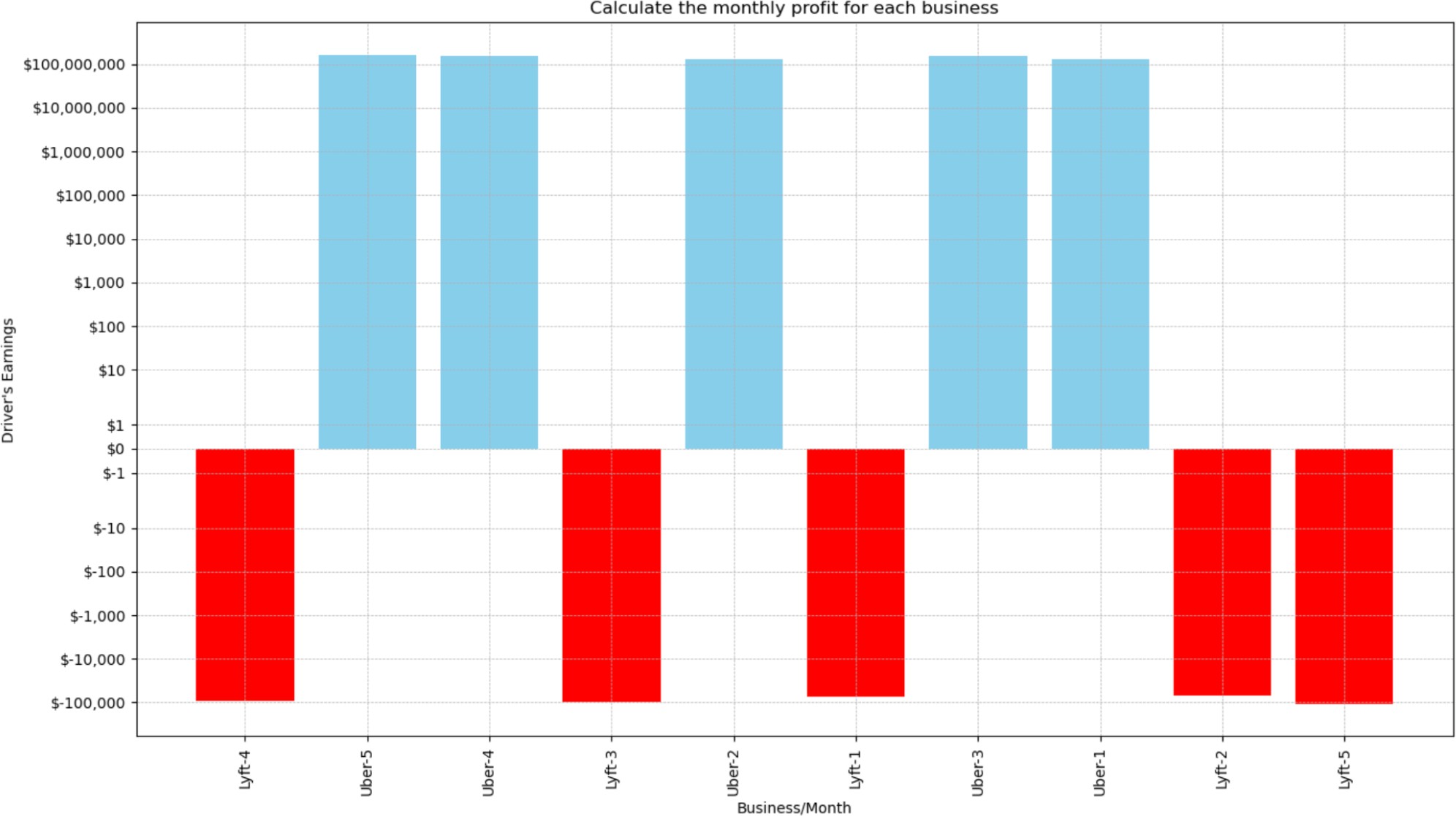
I calculated the monthly profit for each business by summing the “rideshare profit” for each month. Where I convert the 'rideshare\_profit' field to a float type and then summing up the profits for each 'business-month' combination using the .groupBy() and

.sum() functions.

**Visualization/Screenshots:**



**Histogram:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring proper data type conversion for accurate calculations.

**Overcoming Challenges:**

Utilizing appropriate Spark DataFrame functions for data type conversion

I define colors for positive and negative values in the 'profit\_sum' column. If a value is negative, I assign 'red' and for positive values, I choose 'sky blue'. This approach helps me visually distinguish between positive and negative values in my data visualizations, making the analysis clearer.

**Insights:**

Understanding the profitability of each business over time.

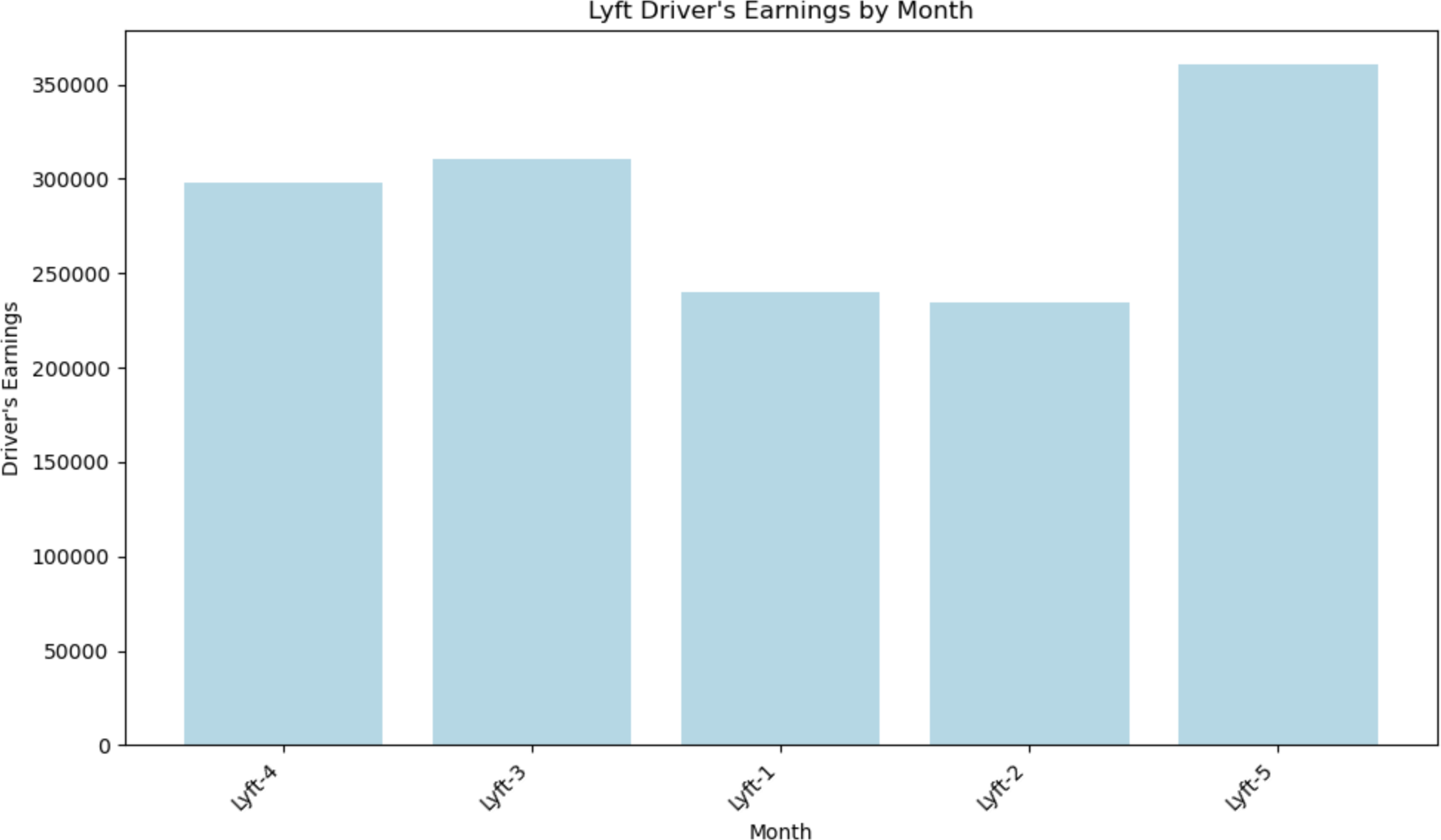
**Task 2(3): Calculate the driver's earnings for each business in each month Explanation:**

I calculated the driver's earnings for each business in each month by summing the driver's total pay. Driver’s earnings were calculated by 'business' and 'month'. Where I converted the 'driver\_total\_pay' field to float for each business.

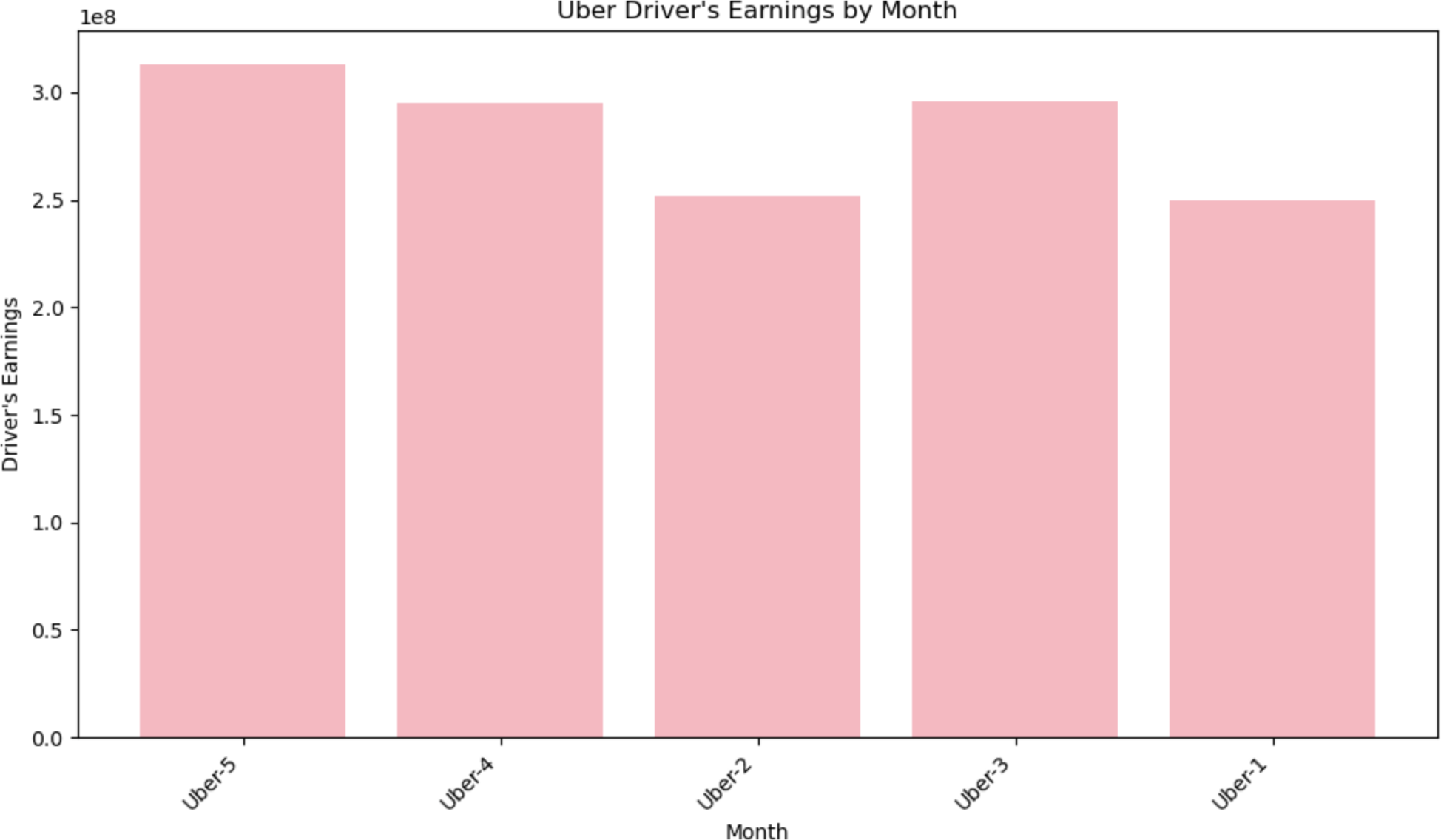
**Visualization/Screenshots:**

**Histogram:**

**For Lyft business Driver’s:**



**For Uber business Driver’s:**



**Challenges/How I overcome them:**

**Challenges:**

Similar challenges to Task 2(2) in terms of data type conversion and data integrity and analysis of given problems.

**Overcoming Challenges:**

I split the data into two business types: one for Lyft and one for Uber. By filtering the 'business\_concat' column to find rows containing 'Lyft' and 'Uber', I create 'lyft\_data' and 'uber\_data' DataFrames, respectively. This separation allows me to analyze and compare the performance and trends of Lyft and Uber independently.

**Insights:**

Identifying variations in driver earnings across different months in both businesses lyft and uber.

**Task 2(4): Extracting Insights and Making Decisions**

After extracting insights the decision can be based on two ways: Monthly profit and Driver’s earning.

**Monthly Profit:** Knowing how your business performs profit-wise every month is crucial to deciding where to invest, who to provide funding to, and how to plan your strategy

**Driver’s Earnings:** Human resource is the important resource for any company success, so employee earning status is directly proportional to productivity encountered in business and hence overall effect on business, it also gives the base to plan the incentives, extra pay etc.

**—------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

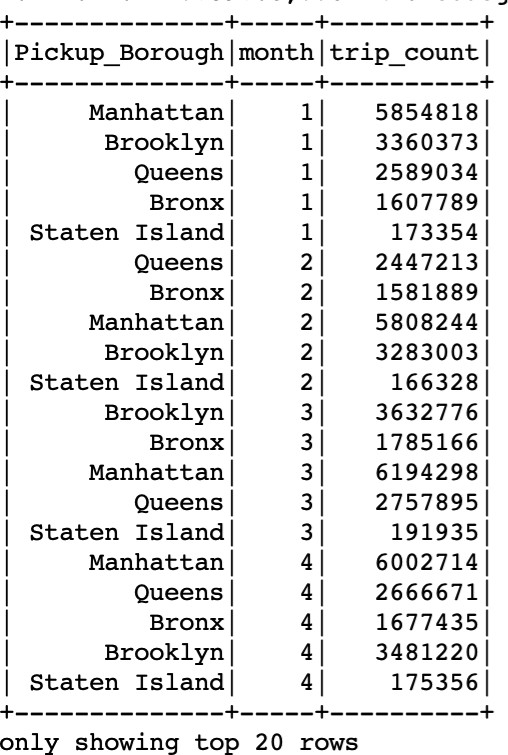
**Task 3 (25 points): Top-K Processing**

**Task 3(1): Identify the top 5 popular pickup boroughs each month Explanation:**

Grouped the data by "Pickup\_Borough" and "month" to count the number of trips for each pickup borough in each month.

groupBy, count, and window functions (Window.partitionBy, orderBy, and row\_number) used to rank and filter the top 5 boroughs by trip count for each month.

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

Implementing window functions to rank rows based on trip counts within each month and handling large datasets efficiently to compute trip counts for each pickup borough in each month.

**Overcoming Challenges:**

implementing window functions by referring to Spark documentation and experimenting with different window specifications until the desired ranking was achieved.

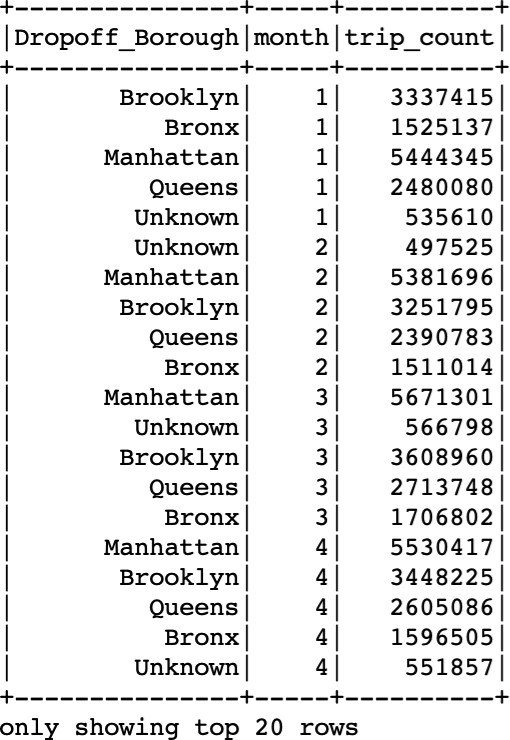
**Insights:**

Identifying the most popular pickup boroughs in each month helps in understanding demand patterns and important locations for rideshare services.

**Task 3(2): Identify the top 5 popular dropoff boroughs each month Explanation:**

Analyzed dropoff boroughs to identify the top 5 popular ones each month, involves grouping by Dropoff\_Borough and month, then counting and ranking the results. same PySpark DataFrame API and window functions were used to analyze dropoff locations as mentioned above

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

'Unknown' as a Dropoff\_borough as input, it could be misleading.

**Overcoming Challenges:**

Applied the same approach used in Task 3(1) for ranking dropoff boroughs by trip count, ensuring consistency in data processing and carefully analyzing the data as some columns include “Unknown”.

**Insights:**

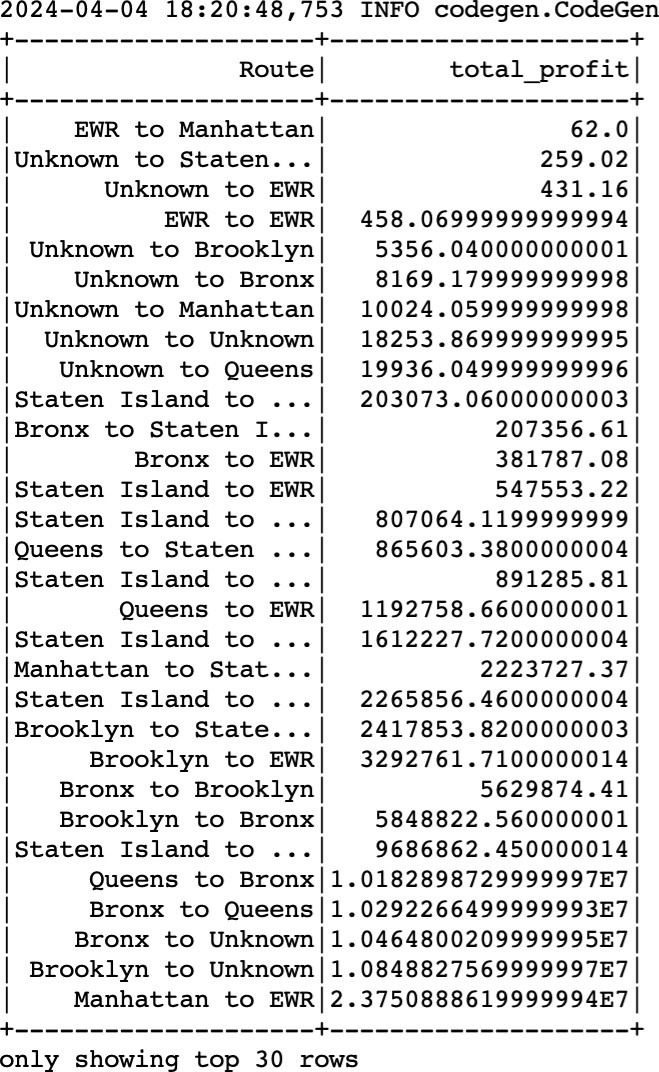
Understanding the most popular dropoff boroughs provides additional insights into travel patterns, destination preferences, and potential areas for service expansion

/optimization.

**Task 3(3): Identify the top 30 earnest routes Explanation:**

Computed the sum of "driver\_total\_pay" for each route (combination of pickup and dropoff boroughs) and selected the top 30 routes based on total profit.

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

Aggregating data at the route level and computing total profits for each route. Handling route combinations and ensuring meaningful identifiers for route analysis.

**Overcoming Challenges:**

Implemented data aggregation techniques to calculate total profits for each route, ensuring accurate computation of driver earnings associated with each route while concatenating pickup and dropoff boroughs.

**Insights:**

Identifying the top profitable routes helps in optimizing driver assignments, pricing strategies, and route planning.

**Task 3(4) Stakeholder Insights and Decisions:**

As a stakeholder ,popular pickup and dropoff boroughs help me to identify demand patterns and customer preferences, guiding resource allocation and marketing strategies.

Analyzing profitable routes enables optimization of driver assignments, pricing structures, and service coverage, leading to improved profitability and customer satisfaction and making decisions to enhance operational efficiency and maximize profitability.

**—------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

**Task 4 (15 points): Average of Data**

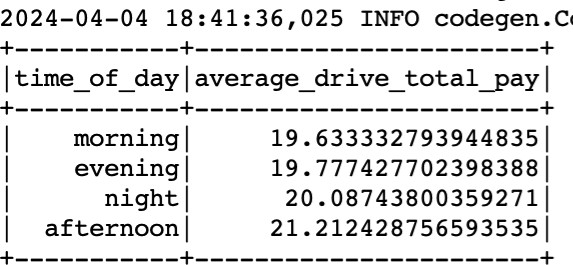
**Task 4(1): Calculate the average 'driver\_total\_pay' during different 'time\_of\_day' periods**

**Explanation:**

Grouped the data by "time\_of\_day" and calculated the average of "driver\_total\_pay" for each time period using groupBy and avg functions.

'average\_drive\_total\_pay' renamed the column, using the withColumnRenamed method and sorted by 'average\_drive\_total\_pay' in descending order to prioritize the time periods with the highest average pay.

**Visualization:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring accurate grouping of data based on distinct time periods

**Overcoming Challenges:**

Data validation checks and careful data preparation steps were implemented to ensure accuracy.

**Insights:**

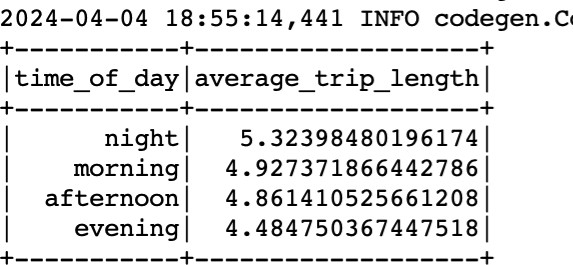
The analysis revealed that drivers earn more during the night and afternoon periods. This insight is invaluable for drivers planning their schedules and for rideshare companies to understand peak demand hours and optimizing driver incentives or bonuses during those periods.

**Task 4(2): Calculate the average 'trip\_length' during different 'time\_of\_day' periods**

**Explanation:**

Similar to Task 4(1), grouped by 'time\_of\_day', and the average 'trip\_length' was calculated for each group and average trip length column was renamed, and the DataFrame was sorted in descending order of 'average\_trip\_length'.

**Visualization/Screenshots:**



**Challenges/How I overcome them: Challenges:**

Ensuring consistent data quality and accuracy in trip length measurements across different time periods.

**Overcoming Challenges:**

careful examination of the dataset ensured the capture of subtle differences.

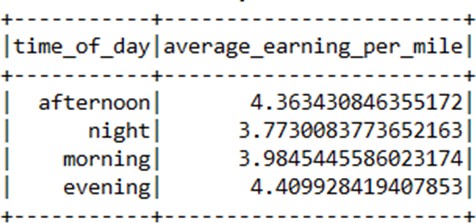
**Insights:**

Night time trips tend to be longer, suggesting the activeness of drivers in the morning rather than at night and hence resulting in optimizing and utilizing time wisely and allocated schedules accordlingly.

**Task 4(3): Calculate the average earned per mile for each 'time\_of\_day' period Explanation:**

joining the average pay and average trip length DataFrames on 'time\_of\_day' to calculate earnings per mile. The average earned per mile was calculated by dividing 'average\_trip\_length' by 'average\_drive\_total\_pay', creating a new column 'average\_earning\_per\_mile'.

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring accurate alignment and synchronization of data from different sources for calculating the average earned per mile.

**Overcoming Challenges:**

careful examination of the dataset to ensure data consistency and accuracy before performing calculations.

**Insights:**

Afternoon and evening times offer higher earnings per mile, suggesting strategic times for drivers to work.

**Task 4(4): Insights and Decisions:**

**Insights from above 3 tasks :**

1. **Earnings v/s Time of Day:** Drivers earn the most on average during the afternoon, followed closely by night and evening periods. This suggests that certain times of day are more profitable
2. **Longer Trip Lengths at Night:** trips are longest on average during the night, which could be due to less traffic or some hidden factors could be there.
3. **Earnings Per Mile in the Afternoon and Evening:** Despite longer trips at night, the afternoon and evening periods provide higher earnings per mile.

**Strategic Implications:**

Companies can use these insights to tailor driver incentives, encouraging more drivers to be available during high-demand periods (afternoon and evening) to meet demand and optimize network efficiency while Understanding how trip lengths and earnings vary by time of day can inform dynamic pricing strategies to balance driver supply and customer demand.

**—------------------------------------------------------------------------------------------------------------------**

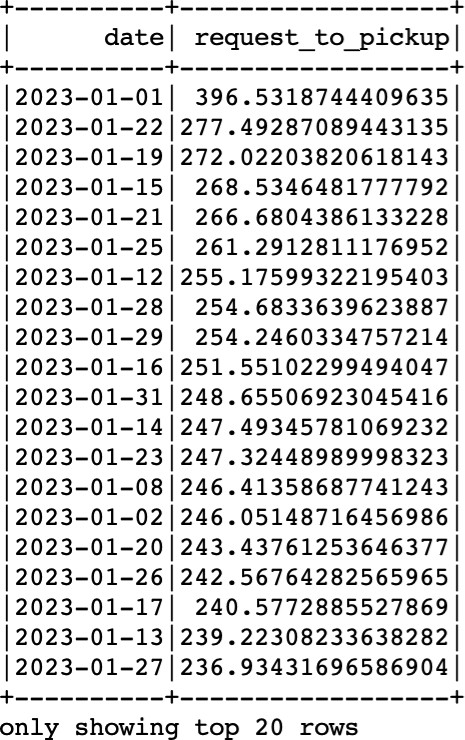
**---------------------------------------------------------------------------------------------------------------------**

**Task 5 (15 points): Finding anomalies**

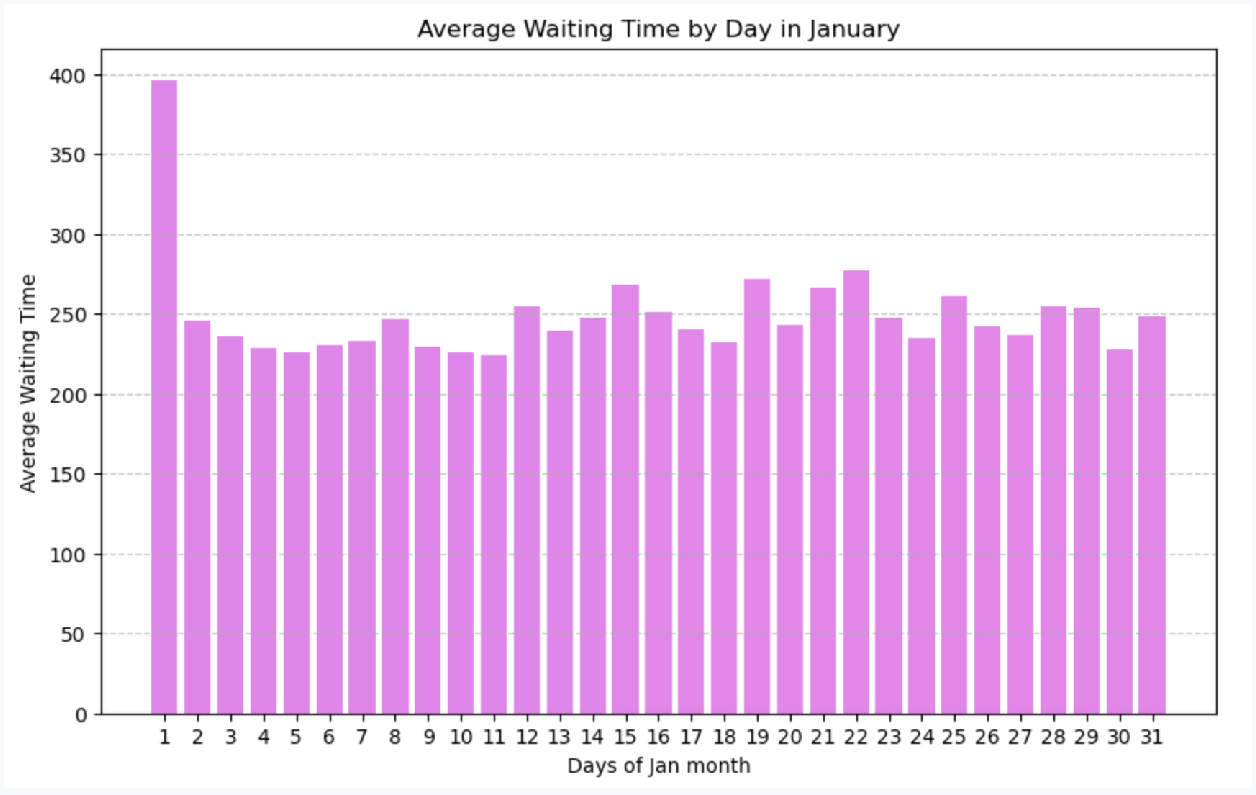
**Task 5(1): Calculate Average Waiting Time in January Explanation:**

Filter the data to include only records from January month and then grouped by the date, and the average (request\_to\_pickup) time is calculated for each day.

**Visualization/Screenshots:**



**Histogram with 'days' on the x-axis and ‘average waiting time’ on the y-axis:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring accurate filtering of data for the specified month.Handling potential inconsistencies in the request\_to\_pickup field that could affect the calculation of average waiting time.

**Overcoming Challenges:**

Data validation and cleaning prior to analysis ensure reliability in the calculated averages.

**Insights :**

The histogram and sorted data reveal that January 1st had significantly higher average waiting times compared to other days, suggesting an anomaly or special circumstance on this date.

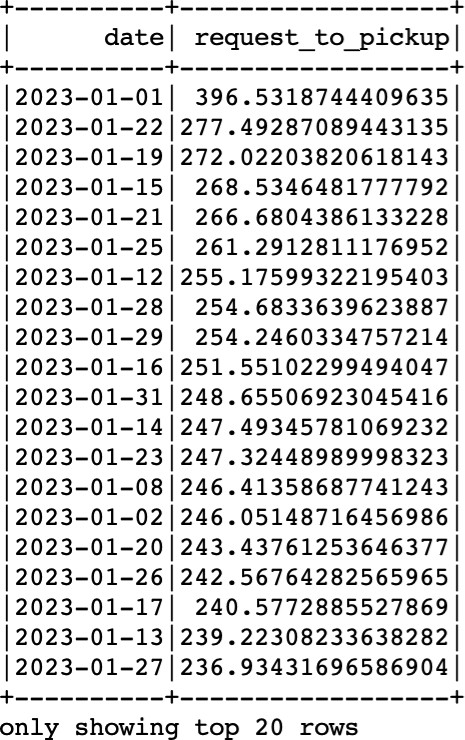
**Task 5(2): Identify Days with an Average Waiting Time Exceeding 300 Seconds**

**Explanation:**

Reviewed the results from Task 5(1) to identify days where the average waiting time exceeds 300 seconds.

From the analysis, January 1st is identified as the day where the average waiting time exceeded 300 seconds, marking it as an outlier.

**Visualization/Screenshots:**



**Challenges/How I Overcome Them:**

**Challenges:**

The calculation depends on the precision of the 'date' and 'request\_to\_pickup' fields. Inconsistent or any small mistake can cost the data analysis and final result.

**Overcoming Challenges:**

Standardize formats for 'date' and 'request\_to\_pickup' fields. Ensure accurate aggregation of daily data.

**Insights:**

Identifying specific days with prolonged waiting times helps in pinpointing potential anomalies or issues in service delivery that require further investigation.

**Task 5(3): Explanation for Longer Waiting Time on Specific Days**

Potential reasons for longer waiting times on specific days could include:

1. Increased demand in ide requests during peak hours as it's a new year
2. Operational challenges such as traffic congestion or adverse weather conditions affect driver availability and response times.
3. Technical glitches impacting delays and route divergence.

**—------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

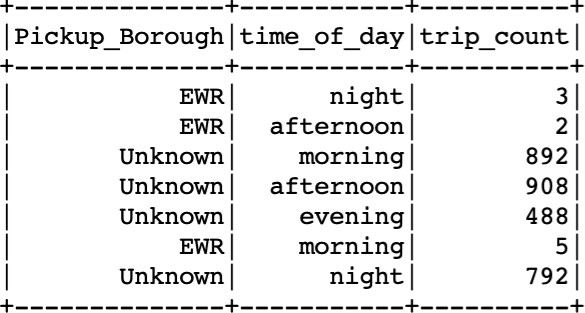
**Task 6 (15 points): Filtering Data**

**Task 6(1) Analyzing Urban Mobility: Trip Count Dynamics by Pickup Borough and Time of Day**

**Explanation:**

First I grouped by 'Pickup\_Borough' and 'time\_of\_day', and the total number of trips in each category was counted. Only groups with a trip count of 1000 or less were kept and renamed to column as 'trip\_count'.

**Visualization/Screenshots:**



**Challenges/How I Overcome Them:**

**Challenges:**

One of the main challenges in handling this kind of data processing task is ensuring that the data is accurately grouped and filtered according to the criteria (in this case, trip counts greater than 0 and less than 1000). This involves a good understanding of operations like groupBy, filter, and withColumnRenamed.

**Overcoming Challenges:**

Basic understanding of Spark operations and ensure it works as expected before applying it to the entire dataset.

**Insight:**

It provide the analysis of number of trips across different boroughs and times of day, less trip count might tell less demand on particular borough location

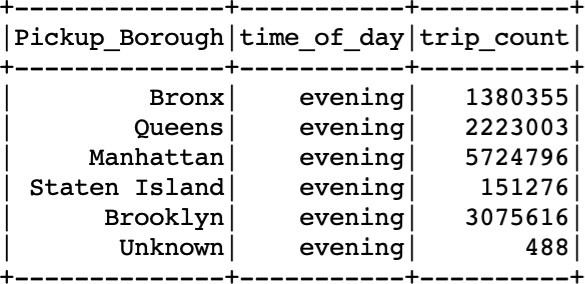
Additionally, the presence of trips in the "Unknown" borough or low trip counts in places like EWR during indicates certain pattern and give space for further investigation

**Task6(2) : Calculate the number of trips for each 'Pickup\_Borough' in the evening time**

**Explanation:**

Similar to Task 6(1), but applied 'evening' as a filter in ‘time\_of\_day’ column and summarized evening trips by 'Pickup\_Borough'.

**Visualization/Screenshots:**



**Challenges/How I Overcome Them: Challenges:**

Ensuring data accuracy and efficiently processing large datasets , faced difficulty in filtering the data with accuracy

**Overcoming Challenges:**

Basic understanding of Spark operations and ensure it works as expected before applying it to the entire dataset and Validate and preprocess data for consistency, define "evening" clearly,

**Insight:**

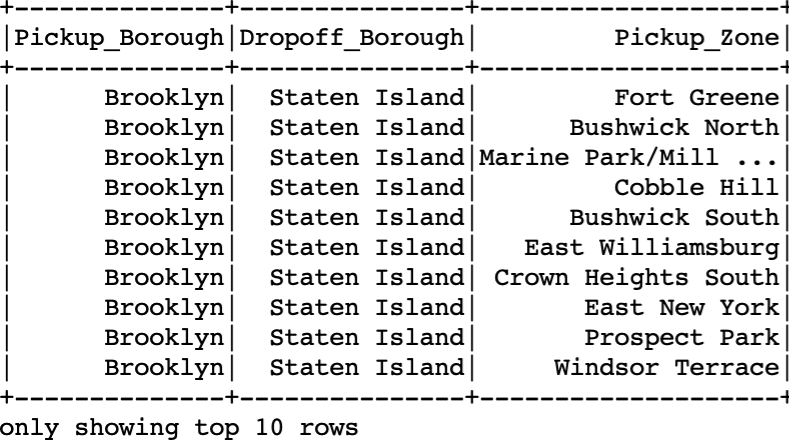
Evening trips are most frequent in Manhattan, followed by Brooklyn and Queens, indicating higher demand in these boroughs during evening hours. Staten Island and areas marked as 'Unknown' have far fewer trips, highlighting potential gaps in service coverage or demand.

**Task6(3) Calculate the number of trips that started in Brooklyn (Pickup\_Borough field) and ended in Staten Island (Dropoff\_Borough field)**

**Explanation:**

The dataset was filtered for trips starting in Brooklyn and ending in Staten Island, showcasing the capability to analyze specific route patterns. This involves grouping the data by Pickup\_Borough, Dropoff\_Borough, and Pickup\_Zone, then applying filters based on the pickup and dropoff.

**Visualization/Screenshots:**



**Challenges/How I Overcome Them:**

**Challenges:**

Handling of large datasets, where performance issues might arise and Efficiently filtering the dataset.

**Overcoming Challenges:**

Careful grouping and using the right methodology to capture the result.

**Insight:**

I observe an interesting pattern between these pickup and dropoff locations which provide insights into commuting trends, potential demands for public transportation.

**—------------------------------------------------------------------------------------------------------------------**

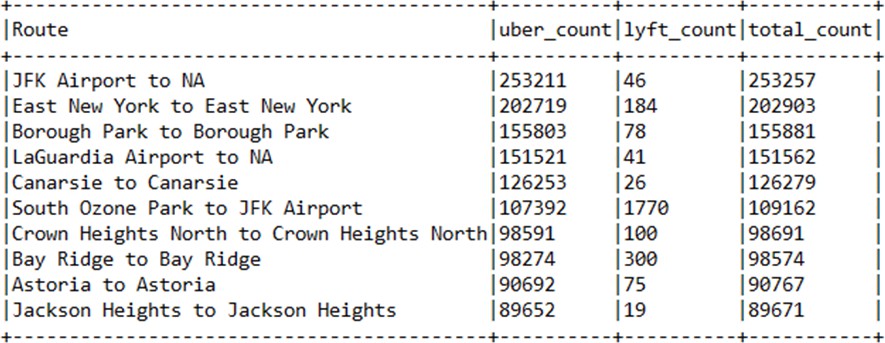
**---------------------------------------------------------------------------------------------------------------------**

**Task 7 (15 points): Routes Analysis Explanation:**

I concatenate 'Pickup\_Zone' and 'Dropoff\_Zone' to form a 'Route' column using the

concat function and and grouped by 'Route' and 'business', and the number of trips (count), separate the counts for Uber and Lyft, a pivot operation is performed on 'business' and changed Columns are renamed for clarity, changing 'Uber' to 'uber\_count' and 'Lyft' to 'lyft\_count' and calculate ‘total\_count’ and print top 10 routes for the same.

**Visualization/Screenshots:**



**Challenges/How I overcome them:**

**Challenges:**

Initially, pivoted DataFrames could contain nulls for routes not served by either Uber or Lyft

**Overcoming Challenges:**

Filling null values with 0 ensures that arithmetic operations (like summing for total counts) proceed without error.

**Insight:**

I observe the highly competition strength between two business and identify the most popular route to optimize anything company wants and also common popular route provise the potential area of travel pattern

**—------------------------------------------------------------------------------------------------------------------**

**---------------------------------------------------------------------------------------------------------------------**

**Task 8 (20 points): Graph Processing**

**Task 8(1): Define StructTypes for Vertices and Edges Explanation:**

I initialization vertexSchema and edgeSchema as StructType objects as shown, where for vertices, the schema should include fields matching the taxi\_zone\_lookup.csv file, typically 'LocationID', 'Borough', 'Zone', and 'service\_zone'. For edges, the schema

represents relationships between vertices, with 'src' and 'dst' fields corresponding to 'pickup\_location' and 'dropoff\_location' from rideshare\_data.csv.

**Challenges/How I overcome them:**

**Challenges:**

The main challenge is understanding the data structure and ensuring the schema accurately represents it.

**Overcoming Challenges:**

To overcome this, one must carefully examine the CSV files and match the schema fields with the data columns.

**Insight:**

This task highlights the importance of accurately defining data schemas in Spark for efficient data manipulation and analysis.

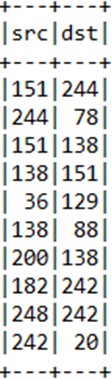
**Task 8(2): Construct DataFrames for Edges and Vertices Explanation:**

edgesDF is defined by extracting 'src' and 'dst' from the rideshare data, and verticesDF

by renaming 'LocationID' to 'id' in the taxi zone data, withColumn() is used to create or replace columns in a DataFrame, here to extract 'src' and 'dst' for the edges DataFrame. withColumnRenamed() renames a column, adapting 'LocationID' to 'id' for vertices.

**Visualization/Screenshots:**





**Challenges/How I overcome them:**

**Challenges:**

A main challenge I faced is to handle large datasets efficiently.

**Overcoming Challenges:**

Using .select() to only process necessary columns or broadcasting smaller DataFrames.

**Insight:**

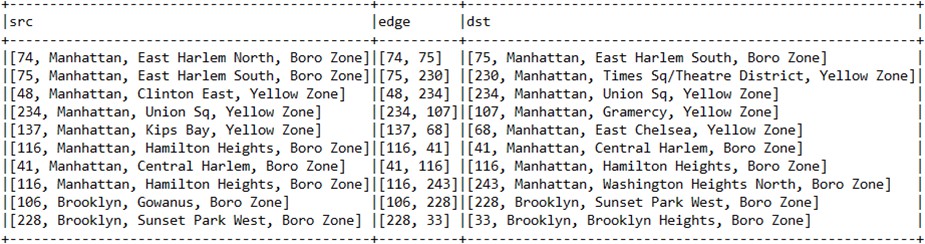
This task emphasizes the importance of preprocessing data to fit the graph model's requirements.

**Task 8(3): Create a Graph and Sample it Explanation:**

This part includes the creation of a graph from the vertices and edges DataFrames and finding relationships (edges) between sources (src) and destinations (dst), showcasing display parts of a graph structure in Spark.

.find() method is then used to explore relationships within the graph.

**Visualization:**



**Challenges/How I overcome them:**

**Challenges:**

Visualizing complex graph structures directly from Spark is challenging.

**Overcoming Challenges:**

Exporting graph data and using dedicated visualization software can overcome this.

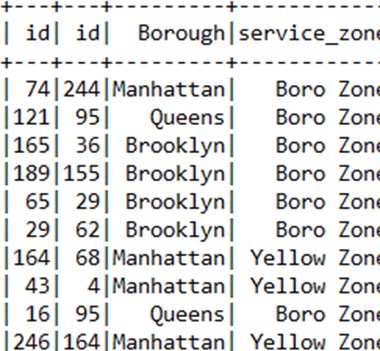
**Insight:**

how to transition from tabular data to graph structures for more complex analyses.

**Task 8(4): Count Connected Vertices with the Same Borough and Service Zone Explanation:**

counts how many times trips start and end within the same area and service zone and join the edges DataFrame with two instances of the vertices DataFrame (differentiate sources and destinations) and filtering for matching 'Borough' and 'service\_zone'.

**Visualization:**



**Challenges/How I overcome them:**

**Challenges:**

Ensuring accurate joins and filters in Spark can be error-prone.

**Overcoming Challenges:**

Careful analysis with smaller datasets can help ensure correctness.

**Insight:**

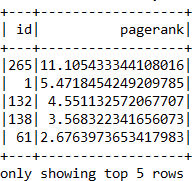
It's an example of combining Spark SQL operations with graph frame operations to analyze relationships within the graph data.

**Task 8(5): Perform PageRank Explanation:**

The PageRank algorithm is applied via PageRank (), with setting specific parameters for the reset probability

This results in a sorted data frame to display the vertices with the highest PageRank scores to identify potentially influential nodes within the graph.

**Visualization:**



**Challenges/How I overcome them:**

**Challenges:**

Tuning the PageRank parameters for meaningful results can be challenging.

**Overcoming Challenges:**

Theoretical understanding of the algorithm's background and experimenting with values to select appropriate values

**Insight:**

Learned the application of graph algorithms in Spark, providing insights into the importance of nodes within a network, which is important for understanding network analysis.